UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/762,944	01/22/2004	Alexander J. Somogyi	ORACL-01337US2	5981

80548 7590 02/04/2009

Fliesler Meyer LLP
650 California Street
14th Floor
San Francisco, CA 94108

EXAMINER	
DAVE, JYOTI D	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
02/04/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/762,944	SOMOGYI ET AL.	
	Examiner	Art Unit	
	JYOTI D. DAVE	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *27 October 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-8 and 10-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-8 and 10-21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *22 January 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *4/21/06 5pg; 3/14/08 2pgs; 10/27/08 1pg.*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 101

1. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 17-21 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

In claims 17, a “first server thread”, a “second server thread”, a “transactional manager”, and a "transactional log" are representative of the software. Therefore, claim 6 has been rejected because it is reasonably interpreted as functionally descriptive material, per se. Software, per se, is not one of the statutory subject matter.

Claims 18-21 are dependent upon claim 17 and are also interpreted as functionally descriptive material.

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made

to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-8 and 10-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Doolittle et al. (6,898,617 B2) in view of Parallel Execution: Oracle8i Concepts Release 8.1.5 (1999) and in further view of Motomura (5,815,727).

In reference to Claim 1:

Doolittle discloses:

A method for implementing a two-phase commit protocol, comprising:

dispatching a first prepare operation from a first server thread to a second server thread (see Fig. 3a, Primary thread pool. See also, Col. 2, liens 50-65, receiving a request [a first prepare operation] to be processed by a thread pool [second server thread])**, wherein the first prepare operation is associated with a first resource and a prepare phase;** (see Col. 5-6, lines 45-67 and 1-10, at a client session a request (i.e. prepare phase) is sent to a thread and a client session is represented by a session data structure, which is used to store information relating to a particular client session).

a second prepare operation by the first server thread (see Fig. 3b, Primary thread pool. See also, Col. 2, liens 50-65, receiving a request [a first prepare operation] to be processed by a thread pool, thread pool selected from a set of one or more eligible thread pools [can be processed by the first thread, while the first prepare operation is processed by a different thread])**, the first prepare operation being processed by the**

second server thread, wherein the second prepare operation is associated with a second resource and the prepare phase (see Col. 5-6, lines 45-67 and 1-10, at a client session a request (i.e. prepare phase) is sent to a thread and a client session is represented by a session data structure, which is used to store information relating to a particular client session).

determining that the prepare phase is complete (see Col. 7, lines 5-15, the state of the request can indicate Complete, specifying that the request completed successfully),

dispatching a first commit operation from the first server thread to a third server thread, wherein the first commit operation is associated with the first resource and a commit phase (see Fig. 3b, Primary thread pool. See also, Col. 2, lines 50-65, receiving a request [a first commit operation] to be processed by a thread pool, Each thread pool includes zero or more available threads to be used by the server in processing an incoming client request [third server thread] [the first commit operation can be viewed as a new request for a thread because the first request, a request for a thread for a prepare phase, is complete and the client would need to create a new request for a commit phase])**;**

a second commit operation by the first server thread in parallel to the first commit operation being processed by the third server thread, wherein the second commit operation is associated with the second resource and the commit phase (see Col.

5-6, lines 45-67 and 1-10, at a client session a request (i.e. second commit operation) is sent to a thread and a client session is represented by a session data structure, which is used to store information relating to a particular client session)**; and**

After determining that the commit phase is complete writing results of the commit phase to a transaction log (see Col. 7, lines 5-15, state of request can include completed, failed or cancelled)**.**

Doolittle does not disclose the operation being **a two-phase commit protocol or the two phase commit protocol associated with a first phase of two phase commit protocol**. However, the article Parallel Execution Oracle8i Concepts Release 8.1.5 discloses a two phase commit protocol operation being executed in a parallel fashion (see page 36, Section
Transaction Model for Parallel DML, subsection Two Phase Commit, A parallel DML operation is executed by more than one independent parallel process transaction. In order to ensure user-level transaction atomicity, the coordinator uses a two phase commit protocol to commit the changes performed by the parallel process transaction). It was well known in the art at the time of the invention that parallel processing could be accomplished through the use of a plurality of threads in parallel (see Motomura, Abstract). Since the article Parallel Execution Oracle8i Concepts Release 8.1.5, discloses a two phase commit protocol operations being executed in a parallel fashion, it would have been obvious to one skilled in the art to execute the parallel processing of

the two phase commit operations using multiple threads since at the time of the invention parallel processing using multiple threads was known and common. It would have also been obvious to one skilled in the art to use the method disclosed in Doolittle to execute the parallel processing using multiple threads because the invention by Doolittle discloses a way of managing thread pools which is more efficient simpler and less expensive than previous approaches (see Doolittle, Col. 2, lines 40-50).

It would also be obvious to one skilled in the art that the two phase commit protocol is associated with the first phase of the two phase commit protocol. It was known at the time of the invention by one of ordinary skill in the art that the two phase commit protocol consists of two phases, the prepare and commit phase. It would be obvious that the two phase commit protocol operation is associated and must execute the first phase in order to complete the two phase commit protocol operation.

Doolittle also does not specifically disclose **processing the operations** in the threads. However, Motomura discloses processing the operations in the threads (see col. 1, lines 25-30, multithreaded programs is executed in parallel per threads on a plurality of processors) This would be an obvious variation because it has been known in the art at the time of the invention that multi-threaded programs can be executed in parallel and would be motivated because the limitations would create a desirable higher performance for the execution of programs (see col. 1, lines 16-19).

In reference to claim 2,

Doolittle discloses:

The method of claim 1 further comprising:

selecting an idle server thread to process the first prepare operation (see Col. 5, lines 44-50, when a client sends a request to a server to be processed, the server obtains an available thread (i.e. an idle thread) from a selected thread pool in order to process the request).

In reference to claim 3,

Doolittle discloses:

The method of claim 2, wherein selecting includes:

determining available server threads in a server (see Col. 6, lines 40-55, the SRB code is used to dispatch the request to an available thread from an eligible pool. Thus, it checks the eligible pools and dispatches the request onto an available service thread of an eligible pool or puts the request on a global queue).

In reference to claim 4,

Doolittle discloses:

The method of claim 3 wherein a thread pool manager determines the available server threads in the server (see Col. 6, lines 40-55, the SRB code is used to dispatch the request to an available thread from an eligible pool. Thus, it checks the eligible pools and dispatches the request onto an available service thread of an eligible pool or puts the request on a global queue).

In reference to claim 5,

Doolittle discloses:

The method of claim 1 further comprising:

reporting results of the prepare phase to a log (see col. 7, lines 5-15, state of request can include completed, failed or cancelled)

In reference to claim 6,

Doolittle discloses:

A method for processing N two-phase commit protocol operations, comprising:

processing N prepare operations in a first server thread, wherein each of the N prepare operations associated with a prepare phase (see Fig. 3a, Primary thread pool. See also, Col. 2, liens 50-65, receiving a request to be processed by a thread pool), **wherein each prepare operation of N-1 of the prepare operations** (see col. 6, lines 5-10, the plurality of thread pools includes a primary thread pool 300 (FIG. 3a) and

a secondary thread pool 302 (FIG. 3b). Each thread pool includes zero or more available threads 304 to be used by the server in processing an incoming client request) **include of operation includes:**

dispatching the prepare operation to a second server thread if a second server thread is determined to be available (see Fig. 3a, Primary thread pool. See also, Col. 2, liens 50-65, receiving a request [a first prepare operation] to be processed by a thread pool [second server thread])**; and**

the prepare operation in the first server thread if no second server thread is determined to be available (see Col. 9, lines 39-50, if no thread is available, the request gets put on a queue and when a thread is available the queue is checked to process the next request)**; and**

N commit operations in a first server thread (see Fig. 3a, Primary thread pool. See also, Col. 2, liens 50-65, receiving a request to be processed by a thread pool [the first commit operation can be viewed as a new request for a thread because the first request, a request for a thread for a prepare phase, is complete and the client would need to create a new request for a commit phase])**, wherein each of the N commit operations are associated with a commit phase, wherein each commit operation of N-1 of the commit operations** (see Fig. 3a, Primary thread pool. See also, Col. 2, liens 50-65, receiving a request to be processed by a thread pool [the first commit

operation can be viewed as a new request for a thread because the first request, a request for a thread for a prepare phase, is complete and the client would need to create a new request for a commit phase]; see also, see Col. 5-6, lines 45-67 and 1-10, at a client session a request (i.e. prepare phase) is sent to a thread and a client session is represented by a session data structure, which is used to store information relating to a particular client session)**, includes:**

dispatching the commit operation to a second server thread if a second server thread is determined to be available (see Fig. 3b, Primary thread pool. See also, Col. 2, lines 50-65, receiving a request [a first commit operation] to be processed by a thread pool, Each thread pool includes zero or more available threads to be used by the server in processing an incoming client request [third server thread] [the first commit operation can be viewed as a new request for a thread because the first request, a request for a thread for a prepare phase, is complete and the client would need to create a new request for a commit phase])**;**

the commit operation in the first server thread if no second server thread is determined to be available (see Col. 9, lines 39-50, if no thread is available, the request gets put on a queue and when a thread is available the queue is checked to process the next request)**;**

a remaining commit operation in the first server thread (see Col. 9, lines 49-55, the request cannot be processed on the thread currently serving the request..redispatch from the secondary pool to the primary pool**; and**

after determining that the commit phase is complete, results of the commit phase are written to a transaction log (see Col. 7, lines 5-15, state of request can include completed, failed or cancelled)**.**

Doolittle does not disclose the operation being **a two-phase commit protocol**. However, the article Parallel Execution Oracle8i Concepts Release 8.1.5 discloses a two phase commit protocol operation being executed in a parallel fashion (see page 36, Section
Transaction Model for Parallel DML, subsection Two Phase Commit, A parallel DML operation is executed by more than one independent parallel process transaction. In order to ensure user-level transaction atomicity, the coordinator uses a two phase commit protocol to commit the changes performed by the parallel process transaction). It was well known in the art at the time of the invention that parallel processing could be accomplished through the use of a plurality of threads in parallel (see Motomura, Abstract). Since the article Parallel Execution Oracle8i Concepts Release 8.1.5, discloses a two phase commit protocol operations being executed in a parallel fashion, it would have been obvious to one skilled in the art to execute the parallel processing of the two phase commit operations using multiple threads since at the time of the invention parallel processing using multiple threads was known and common. It would have also been obvious to one skilled in the art to use the method disclosed in Doolittle to execute the parallel processing using multiple threads because the invention by

Doolittle discloses a way of managing thread pools which is more efficient simpler and less expensive than previous approaches (see Doolittle, paragraph 0010).

Doolittle also does not specifically disclose **processing the operations** in the threads. However, Motomura discloses processing the operations in the threads (see col. 1, lines 25-30, multithreaded programs is executed in parallel per threads on a plurality of processors) This would be an obvious variation because it has been known in the art at the time of the invention that multi-threaded programs can be executed in parallel and would be motivated because the limitations would create a desirable higher performance for the execution of programs (see col. 1, lines 16-19).

In reference to claim 7,

Doolittle discloses:

The method of claim 6 wherein dispatching each two-phase commit protocol operation to a second server thread includes:

determining available server threads in a server (see paragraph 0051, the SRB code is used to dispatch the request to an available thread from an eligible pool. Thus, it checks the eligible pools and dispatches the request onto an available service thread of an eligible pool or puts the request on a global queue)**; and**

selecting one of the available server threads to be the second server thread (see paragraph 0044, when a client sends a request to a server to be processed, the server obtains an available thread (i.e. an idle thread) from a selected thread pool in order to process the request)..

In reference to claim 8,

Doolittle discloses:

The method of claim 7 wherein a thread pool manager determines the available server threads in the server (see paragraph 0051, the SRB code is used to dispatch the request to an available thread from an eligible pool. Thus, it checks the eligible pools and dispatches the request onto an available service thread of an eligible pool or puts the request on a global queue).

In reference to claim 10,

Doolittle discloses:

The method of claim 6 further comprising:

reporting results of the N prepare operation associated with the prepare phase (see paragraph 0054, state of request [prepare phase] can include completed, failed or cancelled).

In reference to claim 11,

Doolittle does not specifically disclose the pool is used for parallel transaction operations. However, Motomura discloses **A dedicated thread pool is used for parallel transaction operations** (see Abstract**).** It would be obvious to one skilled in the art to include the limitations in Motomura with the invention disclosed in Doolittle because Doolittle and Motomura both disclose using multiple threads. Since it was known in the art at the time of the invention, it would be obvious to process the requests in different threads in parallel.

In reference to claim 12,

Doolittle does not specifically disclose a transaction API. However, Motomura discloses a transaction manager (see fig. 10 element 1020). Motomura does not specifically disclose **a transaction manager implements Java Transaction API,** it would be obvious to one skilled in the art to include the Java Transaction API as a transaction manager because the Java Transaction API is known in the art as a standard Java interfaces between a transaction manager and the parties involved in the distributed transaction system.

In reference to claim 13,

Doolittle does not specifically disclose **the first resource is an XA resource.** However, Motomura discloses a transaction manager (see fig. 10 element 1020). Motomura does

not specifically disclose **the first resource in an XA resource,** it would be obvious to one skilled in the art to include the first resource as an XA resource because an XA interface is an industry standard in distributed transaction processing. Motomura discloses distributing processes to different threads for processing. XAresource interface is a Java mapping of the industry standard XA interface.

In reference to claim 14,

Doolittle does not specifically disclose **the first resource is an XA resource.** However, Motomura discloses a transaction manager (see fig. 10 element 1020). Motomura does not specifically disclose **each prepare operation of N-1 of the prepare operations is associated with an XA resource** it would be obvious to one skilled in the art to include the first resource as an XA resource because an XA interface is an industry standard in distributed transaction processing. Motomura discloses distributing processes to different threads for processing. XAresource interface is a Java mapping of the industry standard XA interface. Since XA resource interface is Java mapping, the prepare operation of N-1 of the prepare operation must be associated with the XA resource to map the operation to a thread.

.

In reference to claim 15,

Claim 15 is rejected on the same basis as claim 11.

In reference to claim 16,

Claim 16 is rejected on the same basis as claim 12.

In reference to claim 17,

Doolittle discloses:

A system, comprising.

a first server thread, wherein the first prepare operation by the first server thread, and wherein the first prepare operation is associated with a first resource and a prepare phase (see Col. 5-6, lines 45-67 and 1-10, at a client session a request (i.e. prepare phase) is sent to a thread and a client session is represented by a session data structure, which is used to store information relating to a particular client session)**;**

a second server thread, wherein the first serves thread dispatches a second prepare operation to the second server thread (see Fig. 3a, Primary thread pool. See also, Col. 2, liens 50-65, receiving a request [a first prepare operation] to be processed by a thread pool [second server thread])**, wherein the second prepare operation is associated with a second resource and the prepare phase** (see Col. 5-6, lines 45-67 and 1-10, at a client session a request (i.e. prepare phase) is sent to a thread and a client session is represented by a session data structure, which is used to store information relating to a particular client session) **and the second serves thread to the first prepare operation being processed by the first server thread** (see Fig. 3a, Primary thread pool. See also, Col. 2, liens 50-65, receiving a request [a first

prepare operation] to be processed by a thread pool [second server thread]; see also, Fig. 3b, Primary thread pool. See also, Col. 2, liens 50-65, receiving a request [a first prepare operation] to be processed by a thread pool, thread pool selected from a set of one or more eligible thread pools [can be processed by the first thread, while the first prepare operation is processed by a different thread]);

after the transaction manager determines that the prepare phase is complete (see Col. 7, lines 5-15, the state of the request can indicate Complete, specifying that the request completed successfully)**, a first commit operation associated with the first resource** (see Fig. 3b, Primary thread pool. See also, Col. 2, lines 50-65, receiving a request [a first commit operation] to be processed by a thread pool, Each thread pool includes zero or more available threads to be used by the server in processing an incoming client request [third server thread] [the first commit operation can be viewed as a new request for a thread because the first request, a request for a thread for a prepare phase, is complete and the client would need to create a new request for a commit phase])**, and the first server thread dispatches to a third server thread a second commit operation associated with the second resource** (see Col. 5-6, lines 45-67 and 1-10, at a client session a request (i.e. second commit operation) is sent to a thread and a client session is represented by a session data structure, which is used to store information relating to a particular client session)**, wherein the second commit operation is in the third server thread in the first commit operation** (see Fig. 3b, Primary thread pool. See also, Col. 2, liens 50-65, receiving a request [a first prepare operation] to be processed by a thread pool, thread pool selected from a set of one or

more eligible thread pools [can be processed by the first thread, while the first prepare operation is processed by a different thread])**; and**

a transaction log, wherein after the commit phase is compile, results of the commit phase are written (see Col. 7, lines 5-15, state of request can include completed, failed or cancelled [it would be obvious to write the state because it is already being determined and might be helpful information to clients)**.**

Doolittle does not specifically disclose the pool is used for parallel transaction operations. However, Motomura discloses **A dedicated thread pool is used for parallel transaction operations** (see Abstract**).** It would be obvious to one skilled in the art to include the limitations in Motomura with the invention disclosed in Doolittle because Doolittle and Motomura both disclose using multiple threads. Since it was known in the art at the time of the invention, it would be obvious to process the requests in different threads in parallel.

Doolittle does not specifically disclose a transaction API. However, Motomura discloses a transaction manager (see fig. 10 element 1020). Motomura does not specifically disclose **a transaction manager implements Java Transaction API,** it would be obvious to one skilled in the art to include the Java Transaction API as a transaction manager because the Java Transaction API is known in the art as a standard Java interfaces between a transaction manager and the parties involved in the distributed transaction system.

Doolittle also does not specifically disclose **processing the operations** in the threads. However, Motomura discloses processing the operations in the threads (see col. 1, lines 25-30, multithreaded programs is executed in parallel per threads on a plurality of processors) This would be an obvious variation because it has been known in the art at the time of the invention that multi-threaded programs can be executed in parallel and would be motivated because the limitations would create a desirable higher performance for the execution of programs (see col. 1, lines 16-19).

In reference to claim 18,

Doolittle discloses:

A thread pool manager, wherein the thread pool manager determines available server threads (see col. 6, lines 40-55, SRB code to dispatch the request to an available thread from an eligible pool)

In reference to claim 19,

Doolittle discloses:

Transaction log records results of the prepared phase (see Col. 7, lines 5-15, state of request can include completed, failed or cancelled [it would be obvious to write the state because it is already being determined and might be helpful information to clients).

In reference to claim 20,

Doolittle does not specifically disclose the limitations of claim 20. However, Motomura discloses **All of the prepare operations and all of the commit operations are part of a single transaction** (see col. 1, lines 19-25, multithreaded execution method, one program is divided into a plurality of parallel executable threads by fork operation [all the prepare actions are forked onto different threads]) This would be an obvious variation because it has been known in the art at the time of the invention that multi-threaded programs can be executed in parallel, as seen in Motomura (see col. 1, lines 25-30).

In reference to claim 21,

Doolittle does not specifically disclose the limitations of claim 20. However, Motomura discloses **All of the prepare operations and all of the commit operations are part of a single transaction.** (see col. 1, lines 19-25, multithreaded execution method, one program is divided into a plurality of parallel executable threads by fork operation [all the prepare actions are forked onto different threads and since the prepare operation is finished before the commit operations are initiated, they are both part of the single transaction). This would be an obvious variation because it has been known in the art at the time of the invention that multi-threaded programs can be executed in parallel, as seen in Motomura (see col. 1, lines 25-30).

Response to Arguments

Rejection under Non-Statutory Obviousness-type Double Patenting:

This rejection has been withdrawn in light of the amendments.

Rejection Under 35 U.S.C. 112:

This rejection has been withdrawn in light of the amendments.

Rejections under 35 U.S.C. 103(a):

In reference to claim 1:

Applicant's Arguments: Doolittle does not disclose a thread that performs any processing other than dispatching.

Examiner's Response: However, as stated above, although Doolittle does not specifically disclose processing the operation in the thread, this was commonly known in the art at the time of the invention that processing of the operations could be done in the thread as evidenced by Motomura (col. 1, lines 25-35).

Applicant's Argument: The combination of Oracle 8i does not disclose performing the prepare and commit phase operations in parallel on different parallel execution servers and Motomura is silent on using a two phase commit protocol.

Examiner's Response: Although Oracle 8i does not disclose the two phase commit protocol operation being performed on parallel execution servers, Motomura discloses parallel execution of an operation on multiple threads. Since a two phase commit protocol is an operation that can be performed in parallel, and the two phase commit protocol is an operation, it would be obvious to one skilled in the art to use multiple threads to execute the operation as disclosed in Motomura because both Motomura and Oracle 8i are relating to parallel execution of an operation.

In reference to claim 6:

Applicant's arguments have been considered but are not persuasive.

In reference to claims 2-5 and 7-10:

Applicant's arguments have been considered but are not persuasive.

Conclusion

Applicant's arguments were considered. Accordingly, previous rejections have been withdrawn. New grounds for objections and rejections have been provided. **THIS ACTION IS MADE FINAL**.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jyoti D. Dave whose telephone number is 571-270-

1470. The examiner can normally be reached on 7:30 AM to 5 PM Mon-Fri, Alt Fri. Eastern Time.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Jyoti D Dave/ 1/29/2009
Examiner, Art Unit 2191
/Wei Y Zhen/

Supervisory Patent Examiner, Art Unit 2191